Research in Music and Artificial Intelligence

CURTIS ROADS

Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Although the boundaries of artificial intelligence (AI) remain elusive, computers can now perform musical tasks that were formerly associated exclusively with naturally intelligent musicians. After a historical note, this paper sermonizes on the need for AI techniques in four areas of musical research: composition, performance, music theory, and digital sound processing. The next part surveys recent work involving AI and music. The discussion concentrates on applications in the four areas of research just mentioned. The final part examines how AI techniques of planning and learning could be used to expand the knowledge base and enrich the behavior of musically intelligent systems.

Categories and Subject Descriptors: J.5 [**Computing Applications**]: Arts and Humanities—*music*

General Terms: Design, Experimentation

Additional Key Words and Phrases: Automatic transcription, composition, digital sound processing, intelligent composer's assistant, music performance, music theory, responsive instruments, sound synthesis

INTRODUCTION

> *The operating mechanism [of the Analytical Engine] . . . might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the Engine. Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the Engine might compose and elaborate scientific pieces of music of any degree of complexity or extent.*
>
> —ADA AUGUSTA, COUNTESS OF LOVELACE
> Quoted in *Scientific Memoirs*, Volume 3, R. Taylor, Ed. 1943 [Kassler and Howe 1980]

Several branches of computer science, including interactive computer graphics, digital signal processing, real-time systems programming, computer architecture, and microcoding, extend directly into computer music. In recent years, another branch has begun to have a major impact on the field—*artificial intelligence (AI)*.

AI falls into two broad categories: a scientific side (also known as cognitive science), devoted to the development of theories of human intelligence, and an engineering side (also known as applied AI), devoted to the development of programs that exhibit intelligent behavior, whether of human or nonhuman quality. Music projects have engaged both the scientific and applied aspects.

The precise boundaries of AI are elusive, since they are related to what people feel is "intelligent behavior." In the early 1950s, adding thousands of numbers per second was considered the work of "giant brains." Today's expectations for intelligent behav-

This article is adapted from an invited lecture presented at the annual meeting of the Italian Computer Society, Padua, Italy, October 1982.

CONTENTS

ior are much higher. As computers take on more and more tasks, the borderlines of AI seem to recede. Indeed, to some people "AI" represents what we have not yet achieved, regardless of what has been solved.

A musical case in point of the boundaries of AI is the keyboard transcription problem. In this problem, the musician improvises on a musical keyboard. It is the computer's task to read the keyboard input and produce a printed transcription of the performance in musical notation. A cost-effective implementation of this practical task would be a boon to many publishers and composers.

The low-level part of the keyboard transcription task, that is, recording the time of key depressions and their durations, was developed in the eighteenth century. In the mechanical scheme developed by Père Engramelle, performed keyboard music was registered on paper in a "piano roll" form of notation. This could be manually converted to common music notation by a musician [Leichtentritt 1934]. The German musical engineers J. F. Unger and J. Hohlfeld first implemented such a system—attached to a harpsichord—in 1752 [Boalch 1956; Buchner 1978]. Two centuries were to pass before the development of a computerized keyboard transcription system that used a minicomputer to scan an organ manual, capture keystrokes, and display rudimentary music notation [Ashton 1971; Knowlton 1971, 1972].

A number of such transcription tools have been developed over the past decade [Ben Daniel 1983]. For several years, companies that sell sequencer (note recording) programs and manufacturers of digital sound synthesizers have offered keyboard transcription software to accompany their systems. In 1983, a Japanese manufacturer introduced an $800 electronic keyboard instrument weighing less than 2 kilograms, which offers automatic music transcription with hard copy printed on the same device. This system demonstrates that for simple music played along with a metronome, the keyboard transcription problem is essentially solved.

However, when the complexity of the music to be transcribed is increased by adding a few more voices or more rhythmic complexity, even powerful symbolic computers are brought to their knees by the transcription task. Near-perfect transcription without a metronome is still considered a deep, AI-level problem. In research carried out at M.I.T., musicians were allowed to play freely on a keyboard as the software inferred the meter, tempo, and note durations [Rowe 1975]. The notes were organized into "tempo frames," which were likened to Minsky's *knowledge frames* [Minsky 1975]. One of the AI issues in the transcription of more complicated musical forms is that interpretation and musical judgment are required, because of the inherent ambiguities of common music notation. The machine must not simply transcribe correctly—for there are an infinite number of "correct" but awkward and unreadable transcriptions; it must transcribe into a natural and "musical" notation style. To some degree then, an advanced transcription program must be endowed with

the kind of aesthetic judgment formerly associated only with human musicians.

1. ROADMAP

In the rest of this paper I survey the work of many researchers. First, some tenets of computer music research are discussed, and after a historical note, I deliver a sermon on the need for AI techniques in four areas of musical research:

- composition,
- performance,
- music theory,
- digital sound processing.

The next part of the paper surveys recent work involving AI and music, concentrating on applications that address the four areas of research just listed:

- the intelligent composer's assistant,
- responsive instruments,
- analysis and generative modeling of music,
- recognition and understanding of musical sound.

2. OBSERVATIONS ABOUT COMPUTER MUSIC RESEARCH

Before taking up research in computer music, we need to recognize a few of its main tenets. First, computer music is an interdisciplinary field, combining facets of art, science, and technology. It requires both musical and technical knowledge to appreciate its potential. A substantial body of computer music research has been accomplished, and familiarity with the literature is essential [Roads 1985a, N.d.; Roads and Strawn 1985; Strawn 1985a, 1985b]. Research in computer music is motivated to some extent by its economic potential (e.g., new instruments and tools for the music and recording industries), but also by purely aesthetic, scientific, and technological concerns (e.g., higher sound quality, interesting new musical effects and expressive possibilities, powerful musical tools, better understanding of human musicality).

The musician who does not invest the energy to learn to program will never fully appreciate the possibilities of computers. A grasp of other important technical concepts (e.g., the foundations of acoustics, psychoacoustics, and synthesis techniques) is also important.

Similarly, anyone who would presume to design advanced musical instruments would do well to become familiar with the trends and aspirations of contemporary music. A good starting point would be Edgard Varèse's 1966 article "The Liberation of Sound" [Varèse 1971]. Scientists and engineers should at least accept the concept of an evolving musical tradition and of individual approaches to musical tasks. Musical engineers can provide for these factors in a practical way by designing flexibility and extensibility into what they build. For example, most musicians would welcome the possibility of experimenting with the scales and tunings made possible by digital synthesis hardware. Providing such a capability is a trivial engineering problem. And yet more than one digital synthesizer has been introduced in recent years without a capacity for flexible scale tuning.

In software engineering, the problem of designing representations for music is another area that could benefit from a flexible and extensible approach. The MIDI (Musical Instrument Digital Interface) 1.0 standard, introduced by a consortium of synthesizer manufacturers in 1983 for the exchange of musical information between computers and electronic instruments [International MIDI Association 1983], needlessly restricts musical expression. Music has many layers and dimensions, and AI research on representations has left powerful clues on how to capture this complicated structure.

3. A HISTORICAL NOTE

Behind current interest in AI and music is a long and interesting history. It is worthwhile to cite a few notable events, while foreshortening greatly the description of other developments. (A history of computer music, beginning with the musical automata of antiquity and proceeding through to the present day, is in preparation [Roads

N.d.]. See also Buchner [1978], Leichtentritt [1934], and Simon [1960].)

Published reports trace the construction of musical automata back to the second century B.C. [Buchner 1978]. One of the more significant historical developments was the construction of mechanical carillons in the Netherlands (thirteenth century). These devices were programmable, using a rotating cylinder dotted with holes. Into specific holes the musician inserted wooden pegs, one for each note to be played. The holes acted as a kind of memory, and up to 9000 memory locations were available on some of these carillons. Remarkably, this was centuries before numerical computing devices were invented.

Numerous other mechanical musical automata were made between 1300 and 1910. For example, we know that Leonardo Da Vinci built a mechanical spinet and drum set in the 1500s [Buchner 1978]. Figure 1 is an example of one of the last mechanical devices, a stringed mechanical instrument with percussion, driven by a perforated paper card (1904). Mechanical systems for the performance of music flourished until the twentieth century, when the electrical recording techniques pioneered by Edison and Berliner became dominant.

The composition of music according to specific rules or procedures has a history that extends back a long time before the development of AI [Kirchmeyer 1962]. Recent computer-based experiments were antedated by Guido d'Arezzo's table lookup procedure for assigning vowels to pitches (ca. 1030; see Babb [1978]), by Affligemensis's rules (ca. 1130) along the same lines, and by the musical games of S. Pepys (1670) and W. A. Mozart (ca. 1770). Another important development was D. Winkel's *Componium* (completed in 1821), a mechanical contraption for producing variations on themes programmed into it.

Some mechanical devices for generating music were no more than curiosities, and certainly all of them embodied musical limitations. It would be a mistake, however, to dismiss all such devices on these grounds. Haydn, Mozart, and Beethoven, among others, all composed for finely crafted automatic instruments. It was only

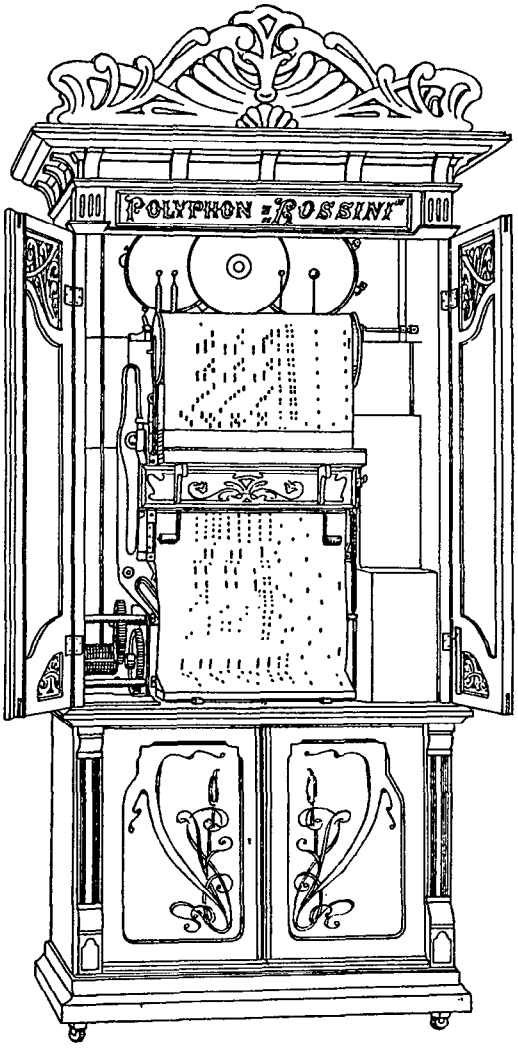


Figure 1. Automatic string instrument with percussion, driven by a perforated paper card (1904).

in the mid-nineteenth century that automatic music devices were manufactured on a commercial basis as mass-produced toys [Ord-Hume 1983]. In music composed before the introduction of the phonograph, existing automatic instruments constitute the only recordings we have of the performance practices from earlier times.

Given this well-documented historical impetus toward mechanism in service of music, it is not surprising that a constant focus of musical research since World War II has been the problem of encoding, gen-

erating, printing, and analyzing music with the aid of computers [Hiller and Isaacson 1959].

4. THE NEED FOR ARTIFICIAL INTELLIGENCE

A main goal of AI is making computers more useful [Winston 1984]. This is an especially pertinent dimension of computer music research, where many obstacles to flexible musician–machine communication still exist. As stated previously, four application areas stand out:

- composition,
- performance,
- music theory,
- digital sound processing.

This section presents the current problems in these areas that appear amenable to solutions offered by AI research.

4.1 Problems in Composition

The current working conditions at many computer music studios remain difficult. Although the number of active composers in the field has increased greatly in the past decade, the paucity of interesting new computer music pieces has been noted by many people. Part of the problem is that it is still hard to compose computer music. It normally requires extraordinary dedication and attention to minutiae. Musicians must learn a set of arcane typewritten protocols for sound synthesis and score input. They must painstakingly design configurations of signal-processing-unit generators and then debug them to make sure they do not violate arbitrary software and hardware restrictions. Since these instruments are driven by numerical data, entering a score into the computer can mean the drudgery of typing in thousands of numbers—up to 20 or more numbers can be required per note.

In the 1950s and 1960s several programs were written for cutting down this work through *automatic composition* [Hiller and Isaacson 1959; Koenig 1970; Moorer 1972; Xenakis 1971; Zaripov 1969], but they all employ a more or less fixed compositional strategy. Interactive systems allowing strategic flexibility are needed.

4.1.1 Problems in Synthesis of Sound

Where software synthesis of sound is employed, work proceeds in nonreal time. For each second of sound, the computer must calculate tens of thousands of *samples*, which represent the sound waveform. It is still necessary for musicians to wait from several minutes to several hours for the sound samples to be computed before they can hear a sound test. At this point, it is common to discover that a minor typing mistake spoiled the entire run. No wonder so many computer music pieces are so short and, despite the computer's vast potential, rely on such a limited timbral palette. The work is too tedious for the musical results.

As real-time synthesizers begin to replace software-based synthesis systems, certain delays associated with sample computation are being alleviated. However, some of the commercial synthesizers embody musical restrictions that are not present in the software synthesis approach, such as the use of a single synthesis technique or even a single pitch tuning.

More important, the problem of weak and one-dimensional musical representations remains, for it is not only the synthesis delay that has hindered creativity. On present systems, none of the higher level structural information within a piece of music is recognized or understood by the computer. (The term "structural" is used in the broadest sense to indicate ways of grouping small musical units into larger musical concepts.) The musical information is stored in the form of numerical note lists, stripped of syntactic and semantic attributes. It is clear that knowledge representation techniques developed in AI research could be useful in capturing this meaningful information, allowing the musician to work directly with it.

4.2 Problems in Performance

Much computer music today is recorded on tape. Performance involves simply playing the tape. Sometimes the composer can ma-

nipulate the volume and tone controls as the tape is played, in order to compensate for the peculiarities of the performance space. Otherwise, the tape performance is fixed.

When performers are to interact with the tape, a tape operator must be present to stop the tape during solos by the performers. When the computer tape starts again, a major problem is synchronizing the performers to the tape. Although it is possible to synchronize the performers by having them wear headphones with a precise "click track" recorded on it, this is an uncomfortable situation for the performers, and it lacks the spontaneity of, say, a chamber music performance. Other solutions have been tried, such as the common trick of ensuring that each point of synchronization is a crescendo with a long buildup, but they lack musical flexibility. A computer system that could use AI techniques to track tempo or listen for other musical cues could bring spontaneity and flexibility to the performance of computer music.

4.3 Problems in Music Theory

Traditionally, music theory's primary objectives include the analysis of musical scores and the construction of generative models for music. More recently, the influence of cognitive science has been seen in the formation of such journals as *Psychomusicology* and *Music Perception*. These journals focus on the processes of hearing and musical cognition, and their contributors are drawn from the scientific community as well as the music community.

Many music scholars have yet to incorporate the computer into their theorizing, despite the computer's obvious utility as a model analysis and testing medium. A handful of musicologists have used computers for theoretical work. Their research has been dominated by statistical techniques [Lincoln 1970]. By far the most prevalent methods of computer analysis have involved simple labeling or tallying of musical surface features in long uninterpreted reports (e.g., Boody and Reidel [1981]). For example, some studies count the number of C-sharps in a composition or cite each occurrence of a minor third interval in an attempt to characterize its musical style. Another technique involves a mechanistic search for common patterns in different pieces in an effort to find "borrowings" from one piece in another [Lincoln 1973].

Such programs are very limited. From the standpoint of theory construction, it would be much more useful if the computer could not just find instances of musical structures, but actually make connections between them and build internal knowledge structures on the basis of what it finds, that is, understand musical structure. In order to accomplish such a goal, better models of musical form and process are needed, as well as deeper analysis tools.

In AI research, the current focus is on *knowledge-based* systems [Minsky 1975], in which searches for patterns in the input data are aided by domain-specific knowledge. This knowledge is used to organize the searching, parsing, and inferential processes involved in understanding. The ability to infer concepts from partial descriptions is lacking in current computer-based models of musical structure. This topic is discussed in more detail in Section 7.

4.4 Problems in Digital Sound Processing

Digital sound editing and processing have been hampered by the crudeness of our current tools for accessing and manipulating acoustic information. At the computer music centers and recording studios that presently have interactive digital sound-editing facilities, the primary means of finding a particular point in the music is to look at a graphic plot of amplitude versus time for each sample. Alternatively, the editor can play the sound backward and forward until the precise editing point is heard.

Although "zooming" over various time scales is supported in some digital sound editors, the zoom is *not* the solution to the problem of providing handles on musical scope and context. Intelligent digital sound-editing techniques offer the possibility of automatic recognition of musical structure, of parsing out individual lines in a polyphonic texture, and of changing the

pitch, starting time, duration, amplitude, articulation, spatial location, or timbre of individual musical notes, lines, phrases, sections, and other musical groupings.

In all four of the cited areas: composition, performance, music theory, and digital sound processing, we could benefit from the use of more musically intelligent systems. The rigid and mechanistic protocols in use today can be replaced by more flowing and multileveled dialogues.

5. DEPTH IN USER INTERFACES

As interactive graphics techniques proliferate, it is becoming more common to see a user interface to a digital music system based on graphic windows and menus [Buxton et al. 1979; Kornfeld 1981; Yavelow 1985]. Although such user interfaces can improve the efficiency of a music system, it is obvious that they are but an intermediate phase of development. Providing a "front end" to a music system does not, in itself, improve on the shallowness of the internal representations.

To get beyond this cosmetic stage, we need to undertake research that will deepen the user interface. Some proposals toward this goal have already been put forth [Laske 1978; Roads 1981]. The members of the SSSP group at Toronto, who developed one of the most elaborate graphical interfaces to a music system, attempted to attack this problem in their proposals for handling *musical scope* [Buxton et al. 1981]. However, they proposed a traditional passive database approach to the problem, which limited the solution. (See their discussion of databases in Section 6.) AI representations for multiple active knowledge sources based on message passing appear more promising [Minsky 1979, 1981]. (A brief comparison between databases and knowledge bases is given in Section 6. For a comprehensive discussion of their relationships, see Brodie and Zilles [1981].)

6. THE QUESTION OF REPRESENTATION

In all of these musical domains, the help of an active symbolic system, encoded with musical knowledge, would be beneficial to the tasks at hand. The core of the technical problem in each application (composition, performance, theory, and signal processing) is the development of the proper internal representations for the musical domain under consideration [Roads 1984].

A number of musicological projects are aimed at encoding musical data in a database for retrieval [Lincoln 1970]. In databases, however, one is usually restricted to encoding passive data in fixed formats. Items that are related in some way are *indexed* in a common file. The interpretation of the *meaning* of this relationship is not resident in the database proper; rather, it is left to sundry user programs with little or no interconnection between them. Moreover, the results of database operations are rarely used to enrich the database itself.

Knowledge representations include not only passive data structures, but also active interconnections that glue pieces of data into compositional or analytical concepts, hypotheses, plans, and other knowledge structures. A knowledge-based system keeps track of the meaning of the material and performs inferences to determine what information is needed for a given task, even if this information has not been explicitly specified. (See Holtzman [1977] for an example of a system that automatically determines the key of a collection of music notes.)

Another characteristic of a knowledge-based system is the ability to view objects from multiple perspectives, for example, viewing a melody in all its transpositions. These two characteristics—inference from partial descriptions and multiple perspectives—are accomplished through procedural specifications embedded in the knowledge base. Procedural specifications make it possible for the system to reason about musical structure and the interrelations among instances of musical concepts.

Active procedural representations have played a major role in AI research [Winograd 1973; Woods 1981a, 1981b], and the procedural view is also natural to music [d'Arezzo, ca. 1030—see Babb 1978; Minsky 1981]. Procedures have always played a role in discussions on the nature of musical structure—witness the centuries-old debate as to whether fugues are forms or processes [Apel 1972]. In recent years, a

procedural approach to the representation of music has been explored by many researchers and composers [Englert 1981; Greussay et al. 1980; Laske 1973a, 1973b; Roads 1981; Schottstaedt 1983; Smoliar 1971, 1974]. Bamberger's research has focused on the way children use procedural representations in learning about musical structure [Bamberger 1975, 1976a, 1976b, 1979].

7. RECENT RESEARCH

The rest of this article surveys four areas of activity that apply artificial intelligence concepts and techniques to music:

- the intelligent composer's assistant,
- responsive instruments,
- analysis and generative modeling of music,
- recognition and understanding of musical sound.

7.1 The Intelligent Composer's Assistant

As the use of interactive graphics techniques begins to enter the computer music domain, there is a corresponding need to rethink carefully the discourse between the musician and the machine. In past computer music composition, most creative effort has been carried out away from the machine. It has been common for the composer to bring a finished score to the computer music studio, type it in, and orchestrate it with some simple instruments. The result has been a body of works with a mechanistic "switched-on" quality, as if these works had really been meant to be played by traditional instruments. Another variation on this approach has been for the composer to design some synthesis instruments and then go off and compose the piece. When the piece is completed, the composer types in the score. The point here is that the computer is being used merely as a combination word processor and player piano, not as a creative compositional medium in which musical imagery is conceived and developed. The situation is similar to the punchcard days of yore, before interactive programming environments, when computer users worked mostly on paper.

Digital synthesizers with sequencers for recording musical keyboard gestures can help, but for general manipulation of compositional architecture, a symbolic representation is needed. The first efforts at providing more interactive symbolic assistance to the composer relied on formal grammars [Holtzman 1981; Jones 1981; Roads 1978]. These systems were useful for the "shorthand" specification of hierarchical musical structures. For example, one symbol could represent a large-scale musical structure. The structures could be sections, subsections, phrases, measures, motives, and individual notes.

The problem with these systems is that musical structure cannot always be segmented into neat hierarchical groupings [Levy 1975; Lidov 1975]. An inspection of much music reveals overlapping contexts, suggesting several possible parsings. If we want to manipulate musical structures that extend beyond strictly hierarchical groupings, more powerful representations are needed.

The intelligent composer's assistant is an attempt to provide an alternative to this situation [Roads 1981]. The overall goal of an intelligent composer's assistant is to support the composer in highly creative phases of composition. This includes the creation of the plan and architecture of a composition, and the encoding of musical material into the *working score*. In order to facilitate smooth interaction, the system should support the widest possible definition of "musical material" by having several representations for musical knowledge.

After having entered some musical material, the composer should be able to manipulate the material, rearranging, modifying, or deleting arbitrary collections of musical objects. (The term "object" is used here in the computer science sense; it can include procedures, data, or both.) Examples of musical objects found in computer music scores include the following:

waveform and control function tables;
voices or patches (instrument labels);
note parameters;
pitches—absolute or relative to another pitch, scale, or tuning;

durations—absolute or relative to another duration;
modes;
scales;
pitch sets;
notes;
chords;
clusters;
arpeggios;
ornaments—trills, appogiaturas, mordents;
melodies;
lines;
clouds of sound;
motives—in pitch, rhythm, timbre, space;
phrases;
measures;
sections;
groupings based on arbitrary attributes, for example, "all notes played by the trumpet."

In a composer's assistant, this collection of concepts must be extensible, according to the composer's needs. Other kinds of musical objects are abstractions of musical structure used in composing or analyzing a piece of music.

In order to manipulate these objects, the composer is provided with a rich collection of *hooks* (accessors) onto the working score. As Figure 2 shows, hooks are software processes with two prongs: One prong attaches to musical objects in the working score, and the other attaches to a *transform*. The transform is an operation applied to the hooked objects such as "delete" or "soften."

Hooks define a notion of musical scope and an access method onto the musical objects, allowing the musician to play with the material in the working score. An important part of the intelligent composer's assistant idea is that these hooks are not all supplied by the composer; they can be constructed by the system as they are needed.

For example, if we issue the command, "Delete all trills played by the saxophone at the end of a phrase," the system will have to find each instance of such an object and delete it. But what if we have not explicitly indicated which notes are trills? Presuming that the system has a basic knowledge of a trill as a concept, it can use its knowledge, together with inference, to determine which notes are affected by the operation.

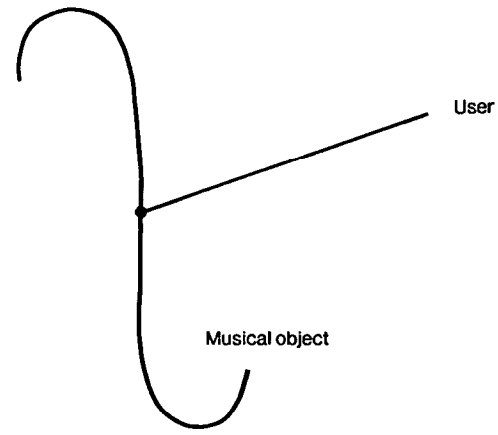


Figure 2. Software hooks for manipulating musical objects.

Another example is a command to "Delete all major seventh chords in first inversion." If the system knows the definitions of common equal-tempered chords, it should be able to find instances of them without further instruction. Many other musical objects can be codified in a similar manner.

Other features of the composer's assistant include graphical interaction, a large catalog of digitized musical sounds, and real-time sound synthesis and processing.

7.1.1 Implemented Systems

One system that incorporates codified knowledge of musical concepts is the Ios system [Roads 1983a]. Ios is not a complete composer's assistant system, but rather a specialized assistant for orchestrating a composer-supplied score. In Ios, the composer specifies rules of orchestration through a series of menus, as shown in Figure 3.

A simple rule looks like this:

(feature instrument).

The Ios program interprets each rule by analyzing the score and searching for instances of musical features named in the rule. Instances of those features are asso-

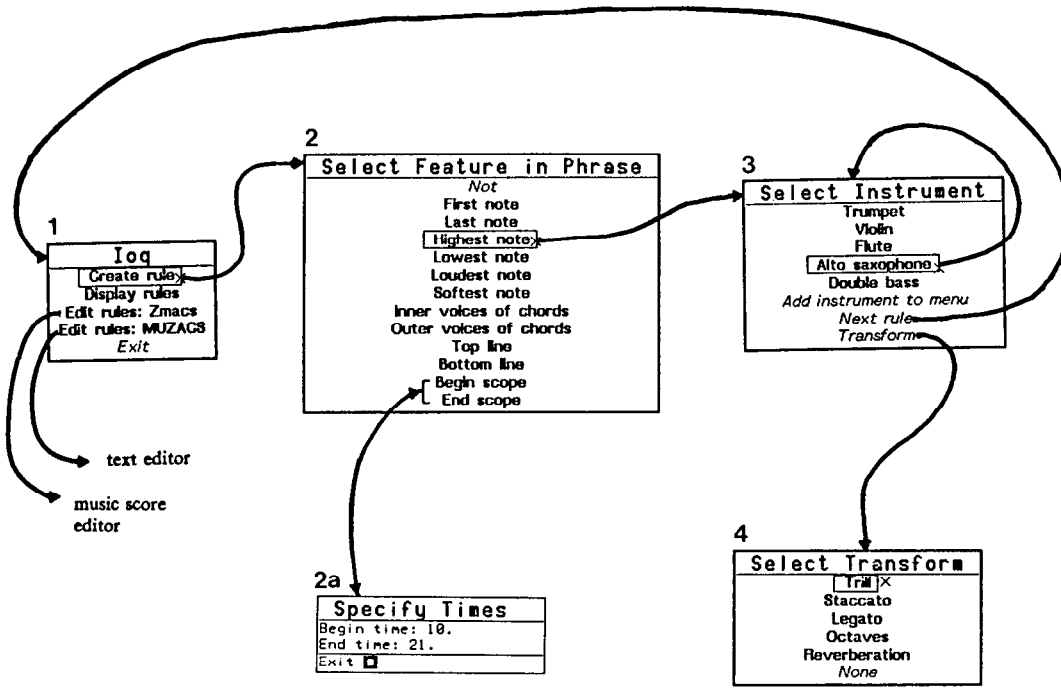


Figure 3. Sequence of menus for interactive orchestration.

ciated with the instruments named in the rules. For example, the rule (*top-line-of-phrase flute*) causes Ios to find the top pitch line in each phrase and associate it with the flute. Transforms such as trills, octave doublings, staccato and legato articulations can also be associated with the rules. Rules in the form of negative constraints ("Do not allow the top line to be played by the trombone") can also be specified, and the scope of rules is variable.

From this research, it is clear that other compositional tasks could benefit from specialized assistant programs. The integration of an interactive graphics interface, music analysis routines, and a variable rule base—all demonstrated within the specialized framework of Ios—also appear applicable to other musical tasks.

Formes is an object-oriented language for music composition and research developed at IRCAM [Cointe 1983; Cointe and Rodet 1984]. Formes is an attempt to provide a framework for integrating models of acoustical and musical knowledge into sound synthesis and analysis. The entities used in digital sound synthesis, such as waveforms, notes, harmonic and rhythmic relations, gestures, and phrases, are defined in Formes as active objects. Each object has the following four components:

name;
environment, in the form (*variable value*);
rules, in the form of LISP expressions;
subobjects, including their environments and rules.

New objects that are similar to or exact replicas of the original object can be cloned. A hierarchy of objects can be built from the bottom up through the subobject concept. For example, the object *Phrase* might have six subobjects corresponding to the six notes included in the phrase.

A unique feature of Formes is the inclusion of an explicitly temporal paradigm into the language. That is, all computation in Formes is synchronized with a dynamic calculation tree that schedules all active objects at appropriate time points. Formes acts as a front-end processor to sound synthesis systems such as Music V [Mathews 1969] and Chant [Rodet et al. 1984].

Another object-oriented system, the Pla language developed at Stanford, contains specialized knowledge that allows composers to develop sophisticated paradigms for voice interaction [Schottstaedt 1983]. A voice object is an integral part of the Pla language. It can be embedded in expressions, be passed parameters, create other voices, read and edit note lists, and serve as a background process that handles multivoice phrasing or scheduling. For example, one mute voice can be used to synchronize several sounding voices. All of the sounding voices ask the mute voice for their ongoing input data (such as their current amplitude value). A mute voice can also start and stop independent voices for a more asynchronous musical structure.

Ios, Formes, and Pla are all designed as specialized systems. In Ios, the main object is the score feature; in Formes, the main object is the phrase; in Pla the main object is the voice. Yet to be developed is a system that integrates several such high-level symbolic representations with more low-level representations such as digital sound files or with flexible graphical notations in a workstation environment. These features, along with the capacity for inference, would lead to a truly powerful composer's assistant system.

7.1.2 Multiple Perspectives on Musical Objects

There are several research issues involved in the design of an integrated intelligent composer's assistant system. First, how can we organize a user interface to support different composing strategies? This begs a more general question: How can the rapid context switching of creative work be best facilitated? In AI research, the "modeless" programming environment of symbolic computers (LISP and Prolog Machines) developed for AI research is a good starting point [Bawden et al., 1979; Sandewall 1978]. In a "modeless" environment (in contrast to a traditional time-sharing environment), it is not necessary to exit one process explicitly, return to a command mode, and then issue a command to enter another process. One can shift from one kind of task to another and back at will. It is possible to examine and manipulate the same object at different levels and in different contexts.

For example, in a music system, we may want to examine a chord object from several perspectives, such as the set of pitches or pitch intervals in it. We may want to hear the chord in all its inversions or transpositions, or display them in common music notation on the screen. Each inversion or transposition represents a different perspective on the same musical object.

Some perspectives allow us to examine the minute details of an object, whereas others put the object into a larger context. For example, we might back off from a chord object to hear the musical context within which that chord is embedded. Looking closer, we might inspect the lowest level representation of the chord to examine and modify the parameter fields within the notes. From this point we might edit the instrument definition that plays the note. Another avenue would be to display the waveform of the sounding chord on the screen and edit it with a pointing device. We could try the same chord with another instrument or assign each note to a different instrument and hear all the permutations. Assuming we hear a permutation we like, we might want to study the spectrum of the chord at different stages to learn more about its evolution or to edit the spectral representation and resynthesize it.

In designing a composer's assistant, another problem is that of designing multiple interacting internal representations, corresponding to musical ideas, for the kinds of cognitive categories used by composers in thinking about their work. For example, a piece may be seen as a function of amplitude in time, a sequence of notes, a series of measures, a set of parallel voices, a collection of phrases, a process of variations on a theme, a three-part sectional form, a five-part form in terms of instrumentation, a process of increasing attack density, a struggle between rhythmic tension and relaxation, and much more—all at once!

7.2 Responsive Instruments

In concert settings, digital synthesizers have taken a place beside tape-recorded performances. Although portable synthe-

sizers are limited in terms of their synthesis capabilities, many musicians appreciate the return to gestural control that these instruments provide. Currently, most commercial synthesizers are equipped with organlike or pianolike keyboards. In effect, this reproduces the performance capability of organs and pianos with different sounds, which satisfies some musicians. However, the possibilities inherent in computer-controlled digital synthesis and processing go far beyond traditional keyboard performance. To begin with, such synthesizers can be controlled by any number and type of input devices, such as buttons, knobs, sliders, batons, drawing tablets, foot pedals, joysticks, mice, breath controllers, guitarlike devices, and other specialized controllers [Buxton 1983]. (At one computer music meeting, it was suggested that a device to track eye movements would be an ideal controller for music, because of the eye's extraordinary muscular precision.) It would be a mistake, however, to suppose that the problem of gestural control of computer music will be solved by exotic input devices alone. Such devices must be connected to the music representations inside the machine in an intelligent way. The engineering problem is to map the data coming from such devices into the control of meaningful musical parameters.

One challenge is to go beyond the kinds of control given to a traditional instrumentalist, such as control of a single voice, to control at the level of a conductor of an ensemble. The key to such a problem is to decode significant gestures and locate the appropriate musical handles at the right grain of resolution for a specific piece. While an instrumentalist in an ensemble controls the pitch, volume, articulation, and timbre of a single voice (or several voices in the case of a polyphonic keyboard), a conductor controls such ensemble parameters as tempo, overall articulation, stress, balance of voices, and spatial projection. See Chadabe [1984] for an example of control at this musical level.

Gestural information can be gathered by the computer through acoustical and visual channels, as well as by physical devices manipulated by the performer. George Lewis's system "listens" to his trombone playing through a microphone connected to the computer [Roads 1985b]. The computer follows the pitch of the melodic line and responds according to rules of improvisation provided by Lewis. In concert, a digital synthesizer controlled by the computer improvises with the live performer. Since group improvisation involves listening as well as playing, the most successful concerts occur when the performer and the computer listen and play as equals, according to Lewis. Recent research by Dannenberg [1984] and Vercoe [1984] explores aspects of a similar problem, that of tracking and accompanying a live performer in the performance of a score known to the computer. Pitch-detector hardware and software are central to such work.

The most dramatic example of a system that can gather information through visual channels is the now famous Wabot-2 robot developed at Waseda University in Japan and reimplemented by Sumitomo for exhibit at the 1985 International Exposition, Tsukuba, Japan [Roads 1986]. This robot is equipped with a high-resolution (2000 by 3000 pixels) video camera and frame buffer. The robot takes 10–15 seconds to completely scan and recognize a musical score and plan all body movements. It then performs the piece, using two hands and two feet, on the digital organ. However, although the robot can recognize a limited number of spoken phrases, it cannot understand musical sound, including the sound of its own performance.

In addition to physical, acoustical, and visual channels, some responsive instruments take in signal data in the form of interrupts and data transfers from other responsive instruments. Closely linked networks of musical computers can be formed to make up a computer band [Bischoff et al. 1978].

With instruments that are expected to respond in an intelligent way, the musical AI problem is to avoid mechanistic behavior. The computer must have an understanding of specific musical contexts, and it must interpret gestures within such contexts. For example, an improvising computer might not react to every short note

in a rapidly played phrase, but it might realize that a short note following a long section of sustained tones signals a change of context. Of course, more sophisticated contextual cues can be programmed.

Another musical performance situation made possible with computer-based instruments is to have several musicians control one large synthesizer. For example, at a performance at the 1982 International Computer Music Conference in Venice, the 4X synthesizer developed at IRCAM was controlled by three performers at the same time [Blum et al. 1983]. Here again, the need for AI techniques is a matter of degree. If the parameters manipulated by the performers are very simple, for example, amplitude and frequency of the notes generated by the system, then AI is probably not needed. However, if the parameters interact in complex ways, the system will need to interpret their context and use musical judgment—behavior that calls for AI tools.

7.3 Analysis and Generative Modeling of Music

A *generative* approach to music theory involves the construction of a computer program to produce a score than conforms to a known musical idiom. Some of the earliest work in this realm was reported in a survey by Hiller [1970]. Programs that generate musical scores can have at least three goals:

- scientific verification of a music theory (simulating a known style);
- producing an object of aesthetic interest (original composition);
- recreational value (colloquially referred to as fun).

In order to *analyze* a musical score by computer, the score must be encoded, and the software has to recognize common musical patterns. Winograd's pioneering work with systemic grammars for the harmonic analysis of musical scores is well known [Winograd 1968]. Less famous is Simon and Sumner's research in combining declarative and procedural descriptions of musical structure [Simon and Sumner, 1968], and Rothgeb's pioneering study at realizing the existing theories of the figured bass [Rothgeb 1968, 1980].

Other projects for computer-based analysis and generative modeling were carried out by Alphonce [1980], Baggi [Buxton 1977], Ebcioglu [1980], Holtzman [1977], Laske [1973a, 1973b], and Smoliar [1971]. (See also Rahn [1980] for a comparison of some of these systems.)

In the 1970s, representations based on formal grammars [Chomsky 1957, 1965] were tested [Baroni 1982; Baroni and Jacoboni 1975, 1978; Lidov and Gabura 1973; Rader 1974; Roads 1978, 1979]. Other generative systems, not yet implemented in software, were proposed by Lerdahl and Jackendoff [1977, 1983] and Snell [1979]. Lerdahl and Jackendoff used rules to assign generative structure labels to parts of existing pieces, while Snell proposed the use of grammars to generate musical fragments exemplifying a known style. One hopes that results of such work can be used in building knowledge representations of musical structure and semantics.

Familiar jazz styles have received due attention. Modeling of jazz improvisation processes has been carried out by several researchers [Arveiller et al. 1976; Fry 1980; Levitt 1981; Ulrich 1977]. Levitt's program produced melodic "solos" for a given chord progression. His program used the *constraints* representation developed in AI research [Sussman and Steele 1981] (see Section 7.3.1).

Smoliar's recent music work [Smoliar 1980] has concentrated on a LISP-like notation for H. Schenker's (1867–1935) theory of harmony. An expert system based on Schenkerian theory has been proposed by Myhill and Ebcioglu [1983]. Their approach is to generate four-voiced chorales in the style of J. S. Bach, using a rule base of approximately 150 expressions in the first-order predicate calculus. The system is meant to generate the chorales from left to right and backtrack until a solution satisfying all the constraints is found.

In contrast to a Schenkerian approach, Meehan [1980] has suggested that Narmour's *implication/realization* theory of tonal harmony [Narmour 1977] could be modeled using the *conceptual dependency*

paradigm of Schank and Ableson [1977]. Laske characterized music analysis as a process of discovery, in which new musical concepts are built out of a primitive set of concepts. In Laske's theory, concepts are modeled as framelike structures [Laske 1984]. Balaban [1981] has proposed a declarative "generalized concept" model of tonal music that combines elements of attribute grammars, semantic networks, and relational databases.

Generative programs have been developed to model the mental activity of both the composer and the listener. Most models to date have attempted to describe listening through a model of perceptual and rational mental processes. Despite the universally acknowledged role of emotional involvement in musical behavior of all kinds, there have been only a few attempts to consider affect as a part of a model of musical cognition [Minsky 1979, 1981; Roads 1980, 1981].

Cognitive psychology research points to the fundamental role played by emotions in motivation, attitude, attention span, memory, and interest—all factors involved in music making and listening [Mandler 1975]. Clearly, a listening model that takes emotion into account will have an advantage over one that misinterprets blatantly emotional gestures, focuses on irrelevant details, or misses the significance of important musical moments. Studies on emotion and music could serve as the starting point for such computational research [Coker 1972; Imberty 1976, 1979; Meyer 1956]. In AI, emotional processes have been studied in models of belief systems [Schank and Ableson 1977], that is, collections of passionately held opinions that strongly influence reasoning and, by extension, musical thinking.

7.3.1 Constraint Representations

Constraints are a promising knowledge representation for music. A constraint is an intrinsically multiple-viewpoint representation. A simple constraint can be visualized as a network of devices connected by wires [Sussman and Steele 1981]. Data values may flow along the wires, and computation is performed by the devices. As an example, a device could check to see whether a pitch value sent to it was equivalent to another pitch value to which it was connected. A device computes only locally available information and places newly derived values on other locally attached wires. Computed values are *propagated* in this way.

In a constraint network, some of the values of the variables in the devices are *dependent* on the values of other variables in other devices. This notion of dependent variables is familiar; it is just the same as in an algebraic equation like $x \times y + z = 3$, where x, y, and z, are dependent. This equation constraint is no more about how to compute x given y and z than it is about how to compute z given x and y. The point is, we can compute the value of any of the dependent variables from this specification.

Constraints are not limited to modeling quantitative relationships. They can be used to represent qualitative semantic dependencies as well. Just as "story problems" may be decomposed into elementary algebra, the essentials of other kinds of relationships can be expressed with constraints. Because variables are defined in several ways simultaneously (i.e., with respect to several different relationships among variables), constraints provide multiple viewpoints on the entities they represent.

As a simple example of a constraint definition of a musical object, let us consider a six-note chord in equal temperament consisting entirely of intervals of two semitones (a chord constructed out of the whole tone scale). For example, starting with the pitch C, the other notes would be D, E, F-sharp, G-sharp, and A-sharp.

Using the constraint representation, we want to be able to manipulate the chord in several ways. First, we would like to transpose the chord by any interval. This is trivial, since the computer simply adds that interval to all the notes. In a slightly more complicated example, we want to be able to tell the computer to play the chord in all its inversions. One way to do this is to choose a new root and compute all the notes of the chord from that root. Finally, we

want to be able to reconstruct the entire chord given any one of its pitches and its note position in the chord. Since the notes are defined by their intervallic relationship to their neighbors, this is easily accomplished from the constraints. The ability of a constraint model to "fill in the blanks" from a partial description of an object is one of its main features. Starting from this example, it is not difficult to imagine constraint representations for musical entities such as scales, modes, melodies, chord sequences, rhythmic progressions, and other objects.

Musical applications of constraints are recorded in the AI literature. Steels [1979] used constraints to reason about tonal structures in music. His system could solve the "passing-chord" problem, which inserts a chord that is harmonically "near" to both its predecessor and successor. The concept of constraints was used by Levitt in organizing a description system for melodies and a set of procedures for melodic improvisation [Levitt 1981].

Levitt's recent paper gives examples of constraint models of a simplified concept of tonal consonance [Levitt 1983]. In his model (simplified for the purposes of explanation), a pitch is consonant with a chord only if the pitch is contained in the chord. A further simplification is that the chords are of two types only: major and minor triads. Thus the pitch E is consonant with the chord C-major because E is a member of the set {C E G}, while the pitch B is dissonant with respect to the C-major triad. In Figure 4a the two pitches at the top are related to the two chords at the bottom via *consonance?* constraints depicted by circles. A *not-equal* constraint exists between the chords. In Figure 4a the network answers the question: "Is the pitch E consonant with the chord C-major? Is the pitch A consonant with the C-major chord? Is the pitch E consonant with the A-minor chord? Is the pitch A consonant with the A-minor chord?" The answers are drawn in the four circles representing the constraints.

In Figure 4b the network is overconstrained. The parts outlined in bold indicate sources of inconsistency. In particular, the pitch A is not consonant with the C-major chord, although it is constrained to be so by the network.

In present systems, each constraint is implemented as a more or less complex program module, while constraint networks constitute large amounts of code. A key to the progress of the constraint paradigm is the development of languages that support the specification and manipulation of constraints [Borning 1979; Steele 1980, 1982].

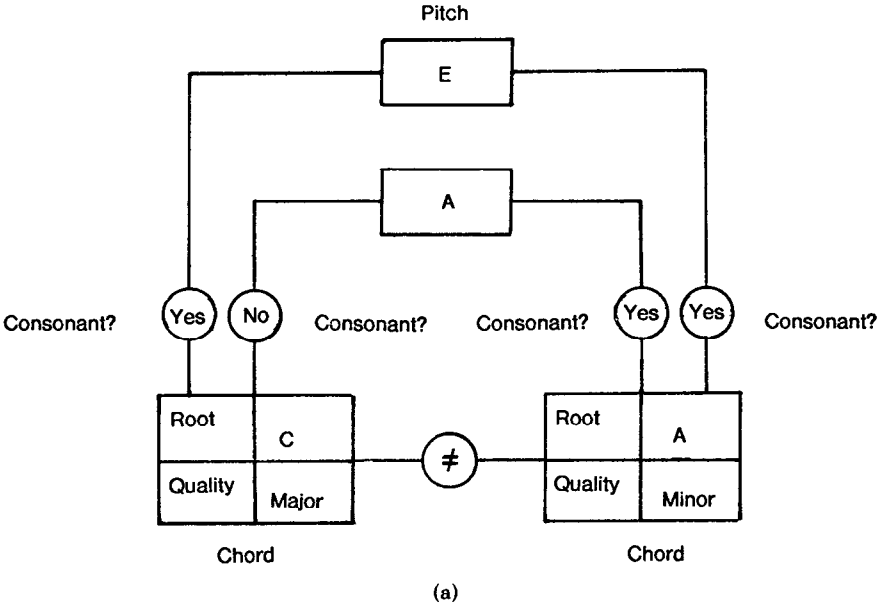
7.4 Recognition and Understanding of Musical Sound

The initial impetus in intelligent signal processing came from speech-understanding research [Erman et al. 1980; Newell 1973]. In signal processing, the distinction between analysis and understanding can be explained as follows. Signal analysis attempts to segment and label an input stream according to preset categories. Signal understanding does not simply categorize a new piece of structure but can also change knowledge structures that already exist. Thus, a main goal of intelligent musical signal processing is building systems that *listen* to and understand music by relating it to previously stored knowledge. Knowledge-based signal processing is also called *signal-to-symbol transformation* in the AI community [Nii and Feigenbaum 1978; Nii et al. 1982].

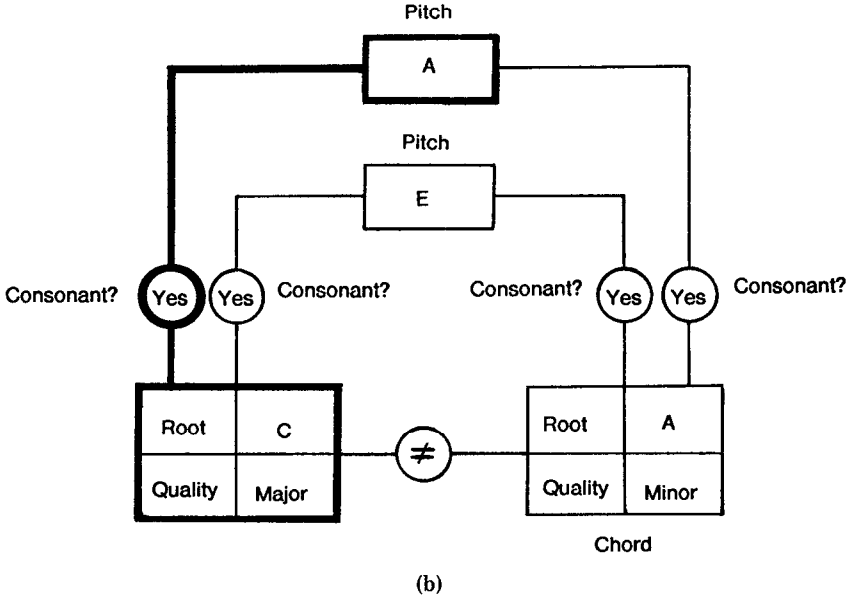
7.4.1 Automated Transcription of Music

A bench mark in the development of a system for the recognition and understanding of musical sound was Moorer's research in programming an automated musical transcription system [Moorer 1975]. Moorer's system interpreted analysis data from a bank of heterodyne filters and built these data into a complete manuscripted transcription of a digital recording of two-part guitar music. The programs inferred notes by accumulating groups of harmonics without combinatorial searching, through the use of knowledge about the internal structure of musical signals.

By maintaining lists of occurrences of various frequencies and eliminating redundant lists (harmonics of already occurring



(a)



(b)

Figure 4. (a) Constraint relation between two pitches and two chords; (b) overconstrained network.

notes), the program obtained a list of regions of specific frequencies. The values on this list were used to set a filter that scanned each region. The filter output indicated strong frequencies, and these data, together with amplitude and duration information, allowed intermediate-level routines to infer which notes were present in the music. Then the program isolated the melody lines and produced a script, used as

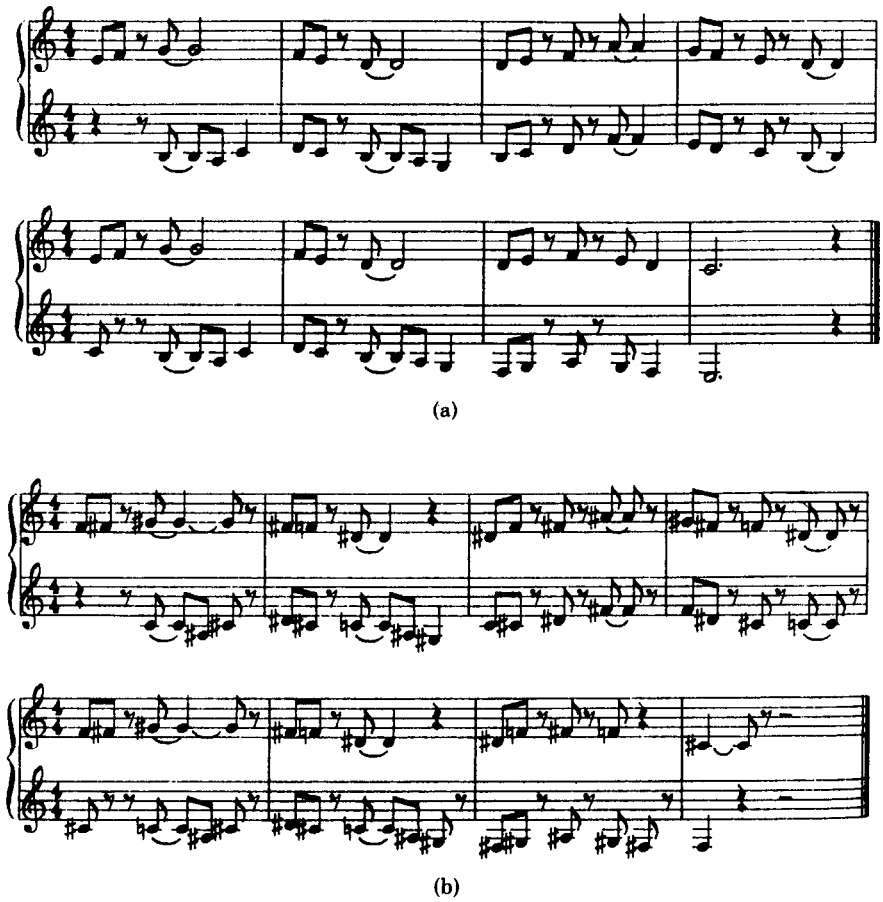


Figure 5. (a) Original score (120 beats per minute); (b) computer-printed score produced by means of automatic music transcription.

input to a music printing program. The result of the entire process was a computer-typeset music score.

Figure 5a shows the original score for a guitar duet. Figure 5b is the score produced by the computer and printed with L. Smith's MSS program [Smith 1973]. The lengths of the longer notes are underestimated because the threshold for noise rejection is set high, cutting the ends of the notes. A note is missing from the last measure. The most conspicuous change was caused by the tuning of the guitar, which was sharp. In Figure 5b the literal-minded computer prints the score one half-step high throughout.

Other systems for sound-to-score music transcription have been implemented by Piszczalski and Galler [1979a, 1979b] and by the Italian researcher Haus [1983]. Haus's system is interesting because it attempted to go beyond traditional music notation to produce a "listening score" for digitally synthesized music.

7.4.2 Intelligent Editors for Musical Sound

One application of AI techniques in signal processing lies in the domain of intelligent editors for musical sound [Chafe et al. 1982; Foster et al. 1982]. Such editors combine signal-processing tools with AI techniques

in order to automatically recognize musically pertinent features of an acoustic signal. In the past a group at Stanford concentrated on the analysis of monophonic classical era works (e.g., fragments of Mozart piano sonatas), but its current research aims at extending the system to account for polyphony and a broader range of musical styles.

The basic tasks of a musical-signal-understanding system are shown in Figure 6. Digitally recorded sound is subjected to acoustic analysis, the results of which are stored in the form of maps of acoustic data. Musical analysis uses the acoustic maps as input to plot maps of the fundamental musical structure.

Temporal analysis is the strong point of the current Stanford system (B. Mont-Reynaud, personal communication, 1983). This analysis is brought to bear on the tasks of tracking tempo, counting beats, determining meter, and assigning note values. Their program separates two subproblems in the analysis of temporal structure:

- The system must express all note values in terms of a coherent musical unit, even when the tempo of that unit fluctuates in performance.
- The system must name that unit, for example, a quarter-note, and determine its relationship to the beat and the measure. Figure 7 shows the structure of the first and hardest problem, that of tempo tracking.

The upper left part of Figure 7 shows the procedures that extract the important events. These events serve as structural anchors in the music. The heuristic applied here is that easily recognized rhythmic or melodic accents normally happen at structurally important points such as strong beats. Thus the duration from anchor to anchor is often a simple relationship. Since this is not always true, the upper right part of Figure 7 shows the procedures that use an independent method of tracking tempo fluctuations. These procedures search for repeating patterns in successive durations and keep running statistics on the most common durations. Significant durations are usually in a simple relationship to one another and to the anchor-to-anchor durations.

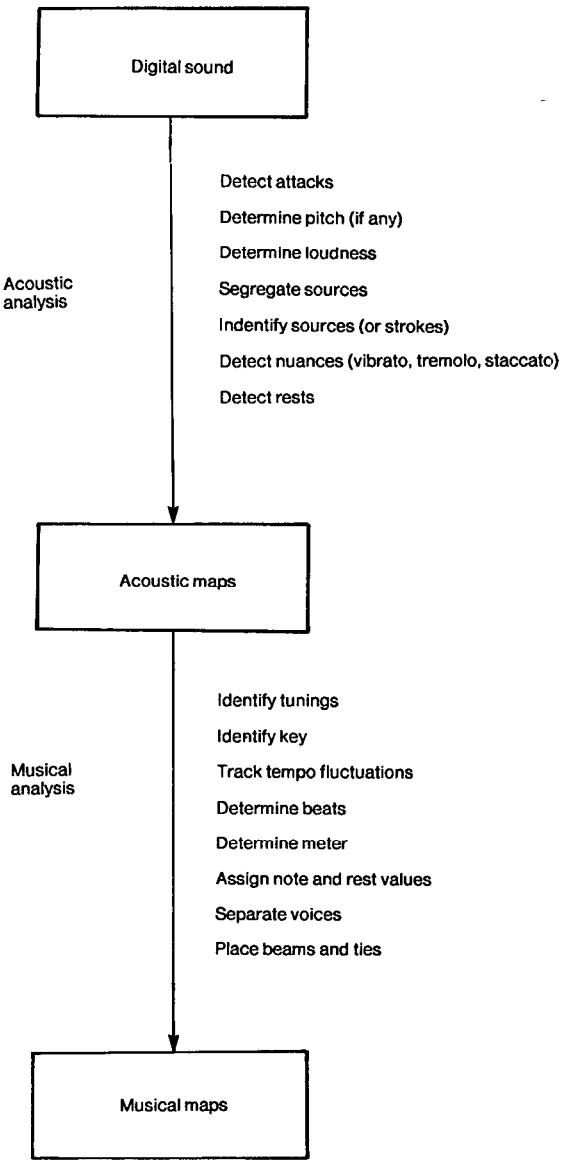


Figure 6. Tasks of musical signal understanding.

By combining these two approaches, tempo tracking decisions select a reasonable hypothesis about the current tempo. The flexibility of the approach is demonstrated in the presence of syncopation—the anchors occur off beat, but the significant durations keep track of the tempo. Conversely, when anchors give strong hints,

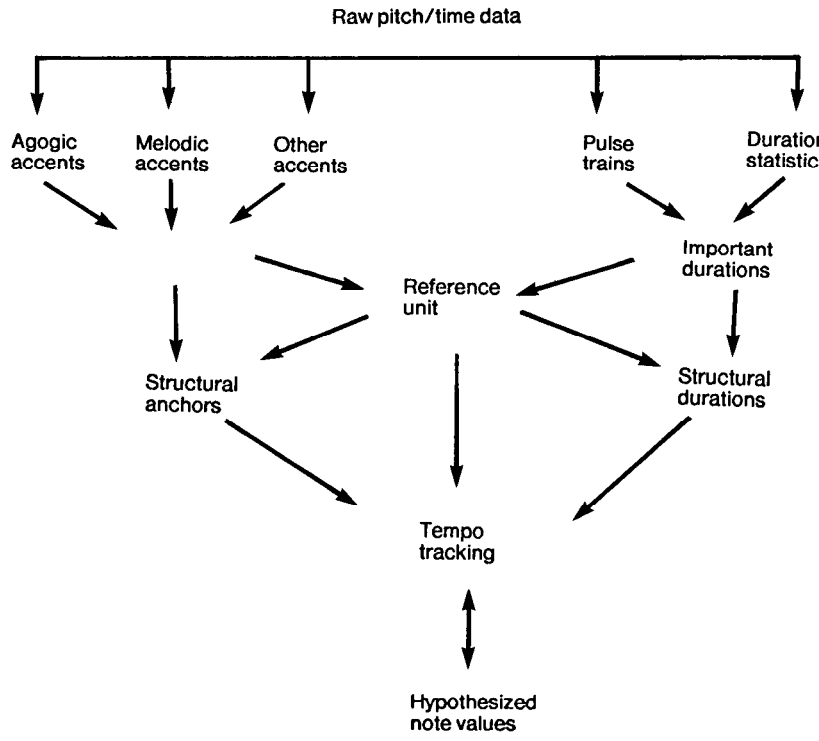


Figure 7. Tempo tracking algorithm.

major tempo adjustments are accommodated.

To date, most researchers have concentrated on the hardest problem, that is, notated transcriptions from a recording of the sound. For this reason, the complexity of the music they could handle has been limited. Using current techniques, even a task such as "start playing at the thirtieth measure" is seen as "barely imaginable" [Chafe et al. 1982, p. 30] because of the amount of processing it would entail. However, even musicians who are unfamiliar with a piece of music find such a task difficult without a score. So far, few have attempted to solve the intermediate task of following a score already entered into the computer. Using the knowledge within the score could significantly cut down the amount of processing involved in finding musical landmarks. This is a requirement for the interactive use of such tools in a recording studio. Using the editor, the sound engineer issues instructions like: "In measure 35, delete the C-sharp played by the trumpet." The machine would be expected to find the note in the recording and delete it.

Signal-to-symbol transformation has been applied to the interpretation of sonograms and other artifacts of acoustical analysis. The aim of such work is to parse complicated acoustical signal inputs. Efforts are aided by software tools for signal processing embedded in a LISP machine environment, such as the Spire system [Roads 1983b]. In Spire, the task of phonetic transcription is aided by a window-and-menu-oriented graphics interface, symbolic computation (e.g., program writing programs), and high-speed numeric processing for analysis and synthesis of acoustic waveforms.

SpireX is an extension to Spire that gives acousticians the opportunity of extracting groups of sounds from a database according to their sonic attributes [Shipman 1983]. Modifications and statistical analyses can be performed on the extracted sounds. Spire and SpireX were developed with speech applications in mind. However, re-

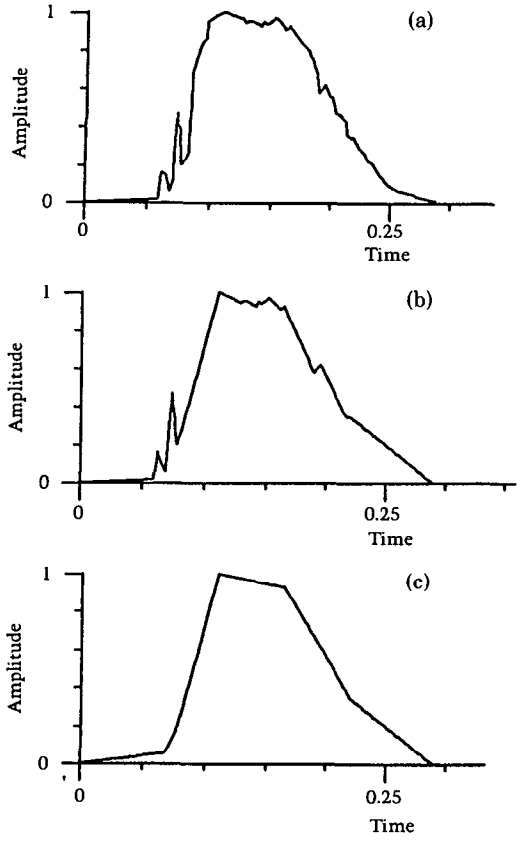


Figure 8. Syntactic hierarchical analysis of amplitude waveforms by computer.

search toward adapting parts of the Spire environment to the entire audio bandwidth has already begun. As with linear predictive coding [Cann 1979, 1980; Moorer 1979] and the phase vocoder [Dolson 1982; Moorer 1978; Portnoff 1976], tools originally developed for speech research can, with some extensions, be highly useful in music applications.

Strawn has recently developed a suite of programs that point toward more intelligent signal understanding [Strawn 1980, 1985c]. Strawn's programs work on the automatic recognition of characteristic envelope shapes in the output of the phase vocoder. The phase vocoder performs a spectral analysis of an incoming sound and produces a set of time-varying frequency and amplitude curves as output. These curves can be used to control a bank of digital sine wave oscillators to resynthesize the sound. Approximation of the amplitude and frequency curves results in enormous amounts of automatic data reduction.

Figure 8 shows an example of automatic syntactic hierarchical analysis of an amplitude waveform, with the resolution set to three successively larger values. Note the effectiveness of the technique as a data reduction method. In Figure 8a, the resolution is the finest of the set; 65 line segments are used to approximate the original signal. In Figures 8b and 8c, 19 and 7 line segments are used, respectively. As a by-product of Strawn's grammar-based approach, the data reduction performed is often of higher quality than that formerly carried out by people.

Another musical application of intelligent signal processors is building responsive instruments for live performance, an area that has already been discussed in Section 7.2.

8. THE FUTURE: PLANNING AND LEARNING

Several areas of AI have just begun to be applied systematically to music, including machine *planning* and *learning*. This section examines the issues involved in these areas.

8.1 Planning in Music Systems

Planning, as usually defined in AI research, is intimately linked with establishing problem-solving strategies. In a problem solver, planning involves setting subgoals dynamically and being able to backtrack and revise subgoals when an attempt to solve a problem fails.

Carrying out any substantial musical task involves planning. For example, instrumentalists make plans for playing a piece by rehearsing and planning optimal fingerings; conductors practice and write comments in scores, and composers have a variety of strategies for realizing their ideas, including the all-important first step of securing a commission. The point is that whenever a variety of problem-solving strategies are possible, planning is used to work out one.

Most music programs tend to cluster around two poles with respect to planning issues. At one pole, past music composition programs pursued set goals using a fixed problem-solving strategy. At the other pole, interactive composition languages and systems can modify musical material, but the tasks given to them are relatively simple (e.g., "delete note") and do not involve planning a sequence of steps or backtracking if a step does not produce the intended result. Also, the larger goals of the composer are not explicitly defined, giving the system no basis on which to plan.

To date, most music systems have tried to deal with very concrete problems of specialized tasks such as sound synthesis or music printing. There has not yet been much attempt to organize work around a common knowledge base that can serve as a foundation for reasoning in and between each of these specialized domains. Only recently has the planning paradigm been incorporated into computer music. Levitt's jazz-oriented system planned the sequence of steps necessary in improvising a melody over a given chord progression [Levitt 1981]. Laske analyzed the human planning behavior necessitated by work with different computer-assisted composition systems such as Xenakis's stochastic music program [Xenakis 1971] and others [Laske 1983].

Planning issues are central to solving certain difficult musical tasks. For example, W. Kornfeld (personal communication, 1983) has suggested that it may be necessary for a program to plan both forward and backward simultaneously in order to construct certain contrapuntal structures. Without such planning the task would be very inefficient. AI research aimed at providing models of temporal plans seems especially applicable to music [Allen and Koomen 1983].

8.2 Learning in Music Systems

Learning has been defined as the process by which a human being or a machine increases its knowledge and improves its skills [Barr et al. 1980; Michalski et al. 1983]. In AI studies, several basic categories of learning programs can be distinguished:

- *Rote learning.* The program memorizes all facts and data.
- *Learning from instruction.* The program transforms knowledge from a concept description language to an internally usable representation. The expressions in the concept description language are provided by an analysis program or by a human teacher.
- *Learning by analogy.* The program transforms and augments existing concepts to understand similar concepts.
- *Learning by example.* The program is presented with a set of "training sequences" that are examples and counterexamples of a concept. The source of the examples is often a human teacher.
- *Learning from observation and discovery.* The program is presented with material. It classifies the material on its own, building new concepts in the process.

A great deal of AI work has focused on learning by example, so it is not surprising that initial studies in music learning by machine also follow this approach. The fundamental techniques of learning by example were developed by Winston [1975]. In Winston's work, a computer program (the student) was presented with examples of concepts in a fictional world of blocks. Figure 9a shows a simple ARCH structure formed out of three bricks, and Figure 9b shows the structural description of the ARCH concept. The square and rectangular nodes are the component objects of the concept, while the labeled arcs are the relations obtaining between the objects. Such a description is typically referred to as a *semantic network*—a widely used knowledge representation [Barr et al. 1980].

Semantic networks can also be used to represent musical concepts, opening the door to musical learning programs along the same lines as Winston's world of blocks. For example, instead of building concepts out of "blocks world" relations like TO-THE-RIGHT-OF, SUPPORTS, ABOVE, and DOES-NOT-TOUCH, we use musical relations like NEXT-EVENT, LAST-

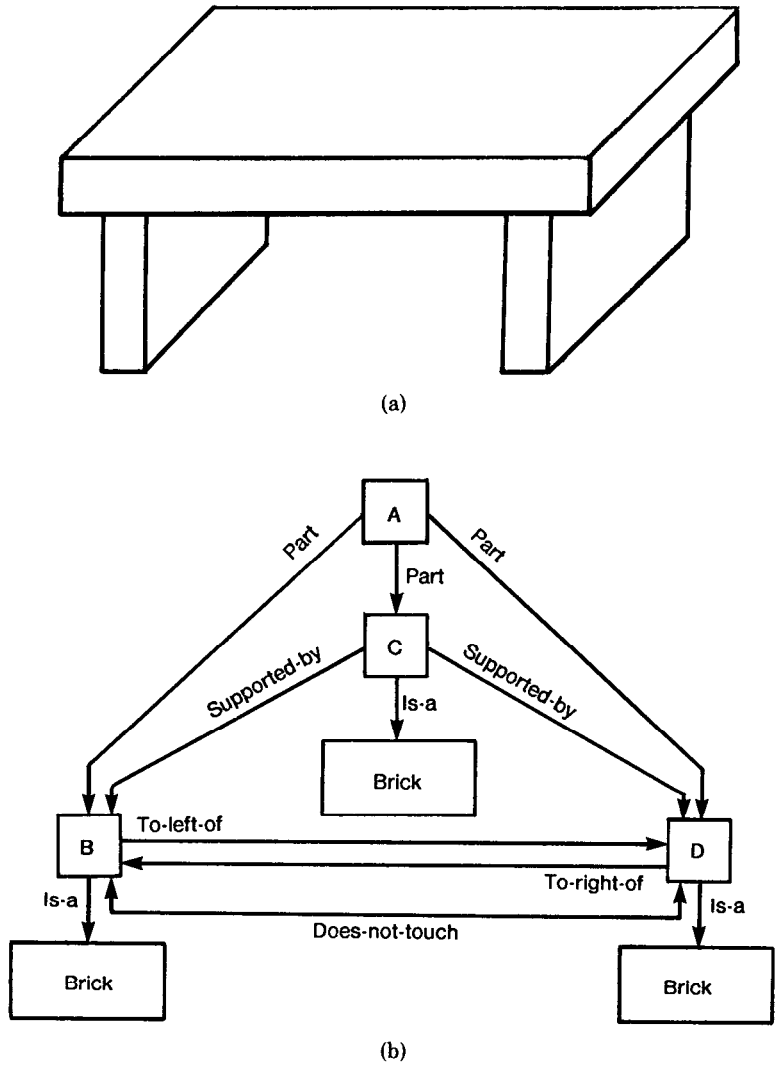


Figure 9. (a) ARCH in the blocks world; (b) structural description of the ARCH concept.

EVENT, NEXT-SCALE-DEGREE, and SAME-CHORD-TYPE.

Figure 10 is an example of a simple musical concept represented as a semantic network. In Figure 10a, a simple five-note phrase ascending the C-major scale is shown. Figure 10b shows a structural description of this phrase.

At Stanford, a group is developing a system for the analysis and understanding of acoustic signals in which learning plays a role (B. Mont-Reynaud, personal communication, 1983). In particular, the group presents training sequences of instrumental sound (e.g., various keys on the piano) to a signal-understanding program. This enables the program to build models of the sounds for later purposes of discrimination, especially in a polyphonic context. At the level of musical structure, the Stanford group is focused on the search for rhythmic and melodic patterns. The group plans a training period for its program during which important patterns are to be pre-

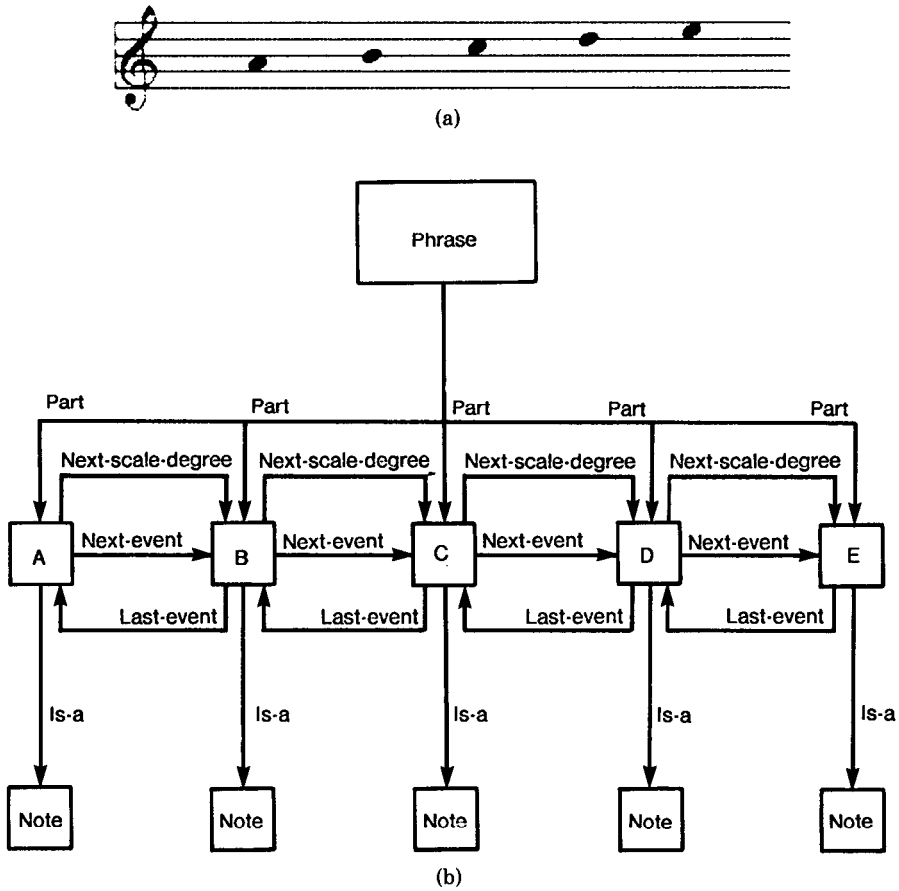


Figure 10. (a) Simple musical phrase; (b) structural description of the simple musical phrase.

viewed before their appearance in a piece. The first such pattern to be tested is arpeggios.

Learning by analogy is another potentially fruitful area for music. The groundwork for learning by analogy was laid by Evans [1968] and Winston [1979, 1980]. Learning by analogy uses procedures to compare the structural descriptions of two different phenomena to extract their pertinent similarities.

Levitt's recent work is based on Evans's analogy program [Levitt 1983]. It generates a structural model of the relationships between a pair of musical objects or their structural descriptions. The system works in the following way. The experimenter gives the program two things: a musical concept description (e.g., a semantic network for describing a melody or a chord progression), and a specific realization of that concept. The realization could be a particular rhythmic setting for the specified melody, or a musical pattern whose underlying harmonic structure mirrored the specified progression. The program then determines the relationships between the concept and the realization. When presented with a second concept description, the program is able to generate its own realization of it according to the first example, because the concept descriptions can be used for generative as well as analytical purposes. Given samples of a particular style (Levitt's domain is traditional New Orleans-style jazz), the program should be able to learn (i.e., be able to recognize and generate) larger scale struc-

tures such as phrases. In this way, the system can be likened to a novice musician who can initially recognize a limited set of cliches and gradually absorb more high-level concepts from samples presented to it.

9. CONCLUSION

> *Allerdings besteht kein Zweifel, dass vieles, was man bisher nur von Berufsmusikern durchschnittlichen Könnens, auf Training aufgebaut, für ausführbar ansah, auch von Automaten geleistet werden kann.* (In any case there is no doubt that much of what average professional musicians can accomplish, based on their training, can also be realized by machines.)
> —K. STOCKHAUSEN
> (1965; in Stockhausen [1978])

The issues raised by AI and music research are indeed broad and deep, combining aspects of the musical arts, science, and technology. Partly because of their multidisciplinary nature and complexity, these issues are also highly interesting! Interesting or not, too little research on artistic applications of technology has been officially supported at academic or industrial laboratories. AI and music research has carried on in spite of this situation, but it has not been reported in the regular AI forums.

As in many fields in which the computer is now entrenched, we cannot expect miraculously quick advances due to AI. Programs that make use of AI techniques are often inherently large and complex; they can take years to develop into useful systems. In the short term, intermediate goals are important. One stepping stone of research in AI and music will be the development of model systems—paradigms that can be studied by many people over a period of time. These paradigms, serving the same sort of common focus as Music V has in sound synthesis research [Mathews 1969] or EMYCIN in expert system design [Barr et al. 1980], could be modified and compared with new systems and conjectures.

REFERENCES

ALLEN, J., AND KOOMEN, J. 1983. Planning using a temporal world model. In *Proceedings of the 1983 International Joint Conference on Artificial Intelligence.* William Kaufmann, Los Altos, Calif.

ALPHONCE, B. 1980. Music analysis by computer: A field for theory formation. *Comput. Music J. 4*, 2, 26–35.

APEL, W. 1972. *Harvard Dictionary of Music.* Harvard University Press, Cambridge, Mass.

ARVEILLER, J., BATTIER, M., AND ENGLERT, G., EDS. 1976. Entry for J. Arveiller. In *Un repertoire d'informatique musicale.* Université Paris VIII, Paris.

ASHTON, A. 1971. Electronics, music, and computers. Tech. Rep. UTEC-CSc-71-117, Computer Science Dept., Univ. of Utah, Salt Lake City.

BABB, W., ED. 1978. *Hucbald, Guido, and John on Music.* Yale University Press, New Haven, Conn.

BALABAN, M. 1981. Toward a computerized analytical research of tonal music. Ph.D. dissertation. Dept. of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel.

BAMBERGER, J. 1975. The development of musical intelligence I: Strategies for representing rhythms. A.I. Memo 342, M.I.T. Artificial Intelligence Lab., Cambridge, Mass.

BAMBERGER, J. 1976a. The development of musical intelligence II: Children's representation of pitch relations. A.I. Memo 401, M.I.T. Artificial Intelligence Lab., Cambridge, Mass.

BAMBERGER, J. 1976b. Capturing intuitive knowledge in procedural descriptions. A.I. Memo 398, M.I.T. Artificial Intelligence Lab., Cambridge, Mass.

BAMBERGER, J. 1979. Logo music project: Experiments in musical perception and design. A.I. Memo 523, M.I.T. Artificial Intelligence Lab., Cambridge, Mass.

BARONI, M. 1982. A project of a grammar of melody. Unpublished manuscript.

BARONI, M., AND JACOBONI, R. 1975. Analysis and generation of Bach's chorale melodies. In *Proceedings of the 1st International Congress on the Semiotics of Music*, G. Stefani, Ed. Centro di Iniziativa Culturale, Pesaro, Italy.

BARONI, M., AND JACOBONI, R. 1978. *Proposal for a Grammar of Melody.* Les Presses de l'Université de Montreál, Montreal, Canada.

BARR, A., FEIGENBAUM, E., AND COHEN, P., EDS. 1980. *Handbook of Artificial Intelligence*, vols. 1–3. William Kaufmann, Los Altos, Calif.

BAWDEN, A., GREENBLATT, R., HOLLOWAY, J., KNIGHT, T., MOON, D., AND WEINREB, D. 1979. LISP machine progress report. In *Artificial Intelligence: An MIT Perspective*, P. Winston and R. Brown, Eds. Addison-Wesley, Reading, Mass., pp. 347–373.

BEN DANIEL, M. 1983. Automated transcription of music. B.Sc. thesis, Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass.

BISCHOFF, J., GOLD, R., AND HORTON, J. 1978. Music for an interactive network of microprocessors. *Comput. Music J. 2*, 3, 24–29. Revised and updated version in *Foundations of Computer Mu-*

sic, C. Roads and J. Strawn, Eds. MIT Press, Cambridge, Mass., 1985.

BLUM, T., ABBOTT, C., AUSTIN, L., BATTIER, M., BEAUCHAMP, J., DASHOW, J., FULLER, W., GROSS, D., HARGE, E., KENDALL, G., LASKE, O., LOY, G., MARC, J., PENNYCOOK, B., POPE, S., AND STRAWN, J. 1983. Report on the 1982 International Computer Music Conference. *Comput. Music J. 7*, 2, 8–35.

BOALCH, D. 1956. *Makers of the Harpsichord and Clavichord, 1440–1840.* Macmillan, New York.

BOODY, C., AND REIDEL, J. 1981. A computer-aided study of Ecuadorean urban music. *Comput. Humanities 15*, 2, 61–74.

BORNING, A. 1979. Thinglab—A constraint-oriented simulation laboratory. SSL-79-3, Xerox Palo Alto Research Center, Palo Alto, Calif.

BRODIE, M., AND ZILLES, S., EDS. 1981. Proceedings of the Conference on Data Abstraction, Databases, and Conceptual Modeling. *SIGART Newsl.* (ACM) *74.*

BUCHNER, A. 1978. *Mechanical Musical Instruments*, I. Irwin, Transl. Greenwood Press, Westport, Conn.

BUXTON, W., ED. 1977. Entry for D. L. Baggi. In *Computer Music 1976/77: A Directory of Current Work.* Commission for UNESCO, Ottawa, Canada.

BUXTON, W. 1983. Continuous hand-gesture driven input. Paper presented at Graphics Interface '83.

BUXTON, W., PATEL, S., SNIDERMAN, R., REEVES, W., PATEL, S., AND BAECKER, R. 1979. The evolution of the SSSP score editing tools. *Comput. Music J. 3*, 4, 14–25.

BUXTON, W., PATEL, S., REEVES, W., AND BAECKER, R. 1981. Scope in interactive score editors. *Comput. Music J. 5*, 3, 50–56.

CANN, R. 1979. Analysis/synthesis tutorial. *Comput. Music J. 3*, 3, 6–11; *3*, 4, 9–13.

CANN, R. 1980. An analysis synthesis tutorial. *Comput. Music J. 4*, 1, 36–42.

CHADABE, J. 1984. Interactive composing. *Comput. Music J. 8*, 1, 22–27.

CHAFE, C., MONT-REYNAUD, B., AND RUSH, L. 1982. Toward an intelligent editor for digital audio: Recognition of musical constructs. *Comput. Music J. 6*, 1, 30–41.

CHOMSKY, N. 1957. *Syntactic Structures.* Mouton, The Hague.

CHOMSKY, N. 1965. *Aspects of the Theory of Syntax.* MIT Press, Cambridge, Mass.

COINTE, J.-P. 1983. *Manuel FORMES.* IRCAM, Paris.

COINTE, J.-P., AND RODET, X. 1984. Formes: Composition and scheduling of processes. *Comput. Music J. 8*, 3, 32–50.

COKER, W. 1972. *Music and Meaning.* The Free Press, New York.

DANNENBERG, R. 1984. An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference, 1984*, W. Buxton, Ed. Computer Music Association, San Francisco.

DOLSON, M. 1982. A tracking phase vocoder and its use in the analysis of ensemble sounds. Ph.D. dissertation, California Institute of Technology, Pasadena.

EBCIOGLU, K. 1980. Strict counterpoint by computer, In *Proceedings of the 1980 International Computer Music Conference*, H. S. Howe, Jr., Ed. Computer Music Association, San Francisco.

ENGLERT, G. 1981. Automated composition/composed automation. *Comput. Music J. 5*, 4, 30–35.

ERMAN, L. D., HAYES-ROTH, F., LESSER, V. R., AND REDDY, D. R. 1980. The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty. *ACM Comput. Surv. 12*, 2 (June), 213–253.

EVANS, T. 1968. A program for a solution of a class of geometric analogy intelligence test questions. In *Semantic Information Processing*, M. Minsky, Ed. MIT Press, Cambridge, Mass.

FOSTER, S., SCHLOSS, W., AND ROCKMORE, A. 1982. Toward an intelligent editor for digital audio. *Comput. Music J. 6*, 1, 42–51.

FRY, C. 1980. Computer improvisation. *Comput. Music J. 4*, 3, 48–58.

GREUSSAY, P., ARVEILLER, J., BATTIER, M., COLERE, C., DALMASSO, G., ENGLERT, G., AND RONCIN, D. 1980. Musical software: Descriptions and abstractions of sound generation and mixing. *Comput. Music J. 4*, 3, 40–47.

HAUS, G. 1983. EMPS: A system for graphic transcription of electronic music scores. *Comput. Music J. 7*, 3, 31–36.

HILLER, L. 1970. Music composed with computers: A historical survey. In *The Computer and Music*, H. Lincoln, Ed. Cornell University Press, Ithaca, N.Y.

HILLER, L., AND ISAACSON, L. 1959. *Experimental Music.* McGraw-Hill, New York.

HOLTZMAN, S. 1977. A program for key determination. *Interface 6*, 29–56.

HOLTZMAN, S. 1981. Using generative grammars for music composition. *Comput. Music J. 5*, 1, 51–64.

IMBERTY, M. 1976. Signification and meaning in music. Monographies de sémiologie et d'analyses musicales III. Univ. of Montreal, Montreal, Canada.

IMBERTY, M. 1979. *Entendre la musique: sémantique psychologique de la musique.* Dunod, Paris.

INTERNATIONAL MIDI ASSOCIATION. 1983. MIDI 1.0 Specification. International MIDI Association, North Hollywood, Calif.

JONES, K. 1981. Compositional applications of stochastic processes. *Comput. Music J. 5*, 2, 45–61.

KASSLER, M., AND HOWE, H. S. 1980. Computer Music. In *Grove's Dictionary of Music and Musicians*, S. Sladie, Ed. Macmillan, London.

KIRCHMEYER, H. 1962. On the historical construction of rationalistic music. *Die Reihe 8*, 11–29.

KNOWLTON, K. 1971. Interactive communication and display of keyboard music. Ph.D. dissertation, Dept. of Electrical Engineering and Computer Science, Univ. of Utah, Salt Lake City.

KNOWLTON, P. 1972. Capture and display of keyboard music. *Datamation 5* (May).

KOENIG, G. M. 1970. Project one. *Electronic Music Reps. 2.* Reprinted 1977 by Swets and Zeitlinger, Amsterdam.

KORNFELD, W. 1981. Everything you always wanted to know about MUZACS but were afraid to grovel through the code to find out. Lecture, M.I.T. Artificial Intelligence Lab., Cambridge, Mass.

LASKE, O. 1973a. Musical semantics: A procedural point of view. In *Proceedings of the 1st International Congress on the Semiotics of Music*, G. Stefani, Ed. Centro di Iniziativa Culturale, Pesaro, Italy.

LASKE, O. 1973b. Toward a musical intelligence system: OBSERVER. *Numus-West 4*, 73, 11–16.

LASKE, O. 1978. Considering human memory in designing user interfaces for computer music. *Comput. Music J. 2*, 4, 39–45.

LASKE, O. 1983. Models of musical planning. Lecture presented at the NEWCOMP Summer Course in Computer Music (Boston, Mass., Aug.).

LASKE, O. 1984. KEITH: A rule system for making music-analytical discoveries. In *Musical Grammars and Computer Analysis*, M. Baroni and L. Callegari, Eds. Leo S. Olschki Editore, Florence, Italy.

LEICHTENTRITT, H. 1934. Mechanical music in olden times. *Musical Quart. 20*, 15–26.

LERDAHL, F., AND JACKENDOFF, R. 1977. Toward a formal theory of tonal music. *J. Music Theor. 21*, 1, 110–171.

LERDAHL, F., AND JACKENDOFF, R. 1983. *A Generative Theory of Tonal Harmony.* MIT Press, Cambridge, Mass.

LEVITT, D. 1981. A melody description system for jazz improvisation. M.S. thesis, M.I.T. Dept. of Electrical Engineering and Computer Science, Cambridge, Mass.

LEVITT, D. 1983. Learning music by imitating. Unpublished manuscript.

LEVY, M. 1975. On the problem of defining musical units. In *Proceedings of the 1st International Congress on the Semiotics of Music*, G. Stefani, Ed. Centro di Iniziativa Culturale, Pesaro, Italy.

LIDOV, D. 1975. On musical phrase. Monographies de semiologie et d'analyse musicales. Groupe, de recherches en semiologie musicale, Univ. of Montreal, Montreal, Canada.

LIDOV, D., AND GABURA, J. 1973. Toward a formal theory of melody. *Comput. Humanities 4*, 3/4, 138–148.

LIEBERMAN, H. 1982. Machine Tongues IX: Object-oriented programming. *Comput. Music. J. 6*, 3, 8–21.

LINCOLN, H., ED. 1970. *The Computer and Music.* Cornell University Press, Ithaca, N.Y.

LINCOLN, H. 1973. Uses of the computer in music composition and research. In *Advances in Computers*, M. Rubinoff, Ed. Academic Press, Orlando, Fla.

MANDLER, G. 1975. *Mind and Emotion.* Wiley, New York.

MATHEWS, M. 1969. *The Technology of Computer Music.* MIT Press, Cambridge, Mass.

MEEHAN, J. 1980. An artificial intelligence approach to tonal music theory. *Comput. Music J. 4*, 2, 60–65.

MEYER, L. B. 1956. *Emotion and Meaning in Music.* University of Chicago Press, Chicago.

MICHALSKI, R., CARBONELL, J., AND MITCHELL, T. EDS. 1983. *Machine Learning.* Tioga, Palo Alto, Calif.

MINSKY, M. 1975. A framework for representing knowledge. A.I. Memo 306, M.I.T. Artificial Intelligence Lab., Cambridge, Mass.

MINSKY, M. 1979. The Society Theory of Thinking. In *Artificial Intelligence: An MIT Perspective*, vol. 1, P. Winston and R. Brown, Eds. MIT Press, Cambridge, Mass.

MINSKY, M. 1981. Music, mind, and meaning. *Comput. Music J. 5*, 3, 8–44.

MOORER, J. A. 1972. Music and computer composition. *Commun. ACM 15*, 2 (Feb.), 104–113.

MOORER, J. A. 1975. On the segmentation and analysis of continuous musical sound. Rep. STAN-M-3. Dept. of Music, Stanford Univ., Stanford, Calif.

MOORER, J. A. 1978. Use of the phase vocoder in computer music applications. *J. Audio Eng. Soc. 26*, 1/2, 42–45.

MOORER, J. A. 1979. The use of linear prediction in computer music applications. *J. Audio Eng. Soc. 27*, 3, 134–140.

MYHILL, J., AND EBCIOGLU, K. 1983. An expert system for Schenkerian synthesis of chorales in the style of J. S. Bach. Unpublished manuscript.

NARMOUR, E. 1977. *Beyond Schenkerism.* University of Chicago Press, Chicago.

NEWELL, A., NEWELL, A., BARNETT, J., GREEN, C., KLATT, D., LICKLIDER, J., MUNSON, J., REDDY, R., AND WOODS, W. 1973. *Speech Understanding Systems.* North-Holland, Amsterdam.

NII, H., AND FEIGENBAUM, E. 1978. Rule-based understanding of signals. In *Pattern-Directed Inference Systems*, D. Waterman and F. Hayes-Roth, Eds. Academic Press, Orlando, Fla.

NII, H., FEIGENBAUM, E., ANTON, J., AND ROCKMORE, A. 1982. Signal-to-symbol transformation: HASP/SIAM case study. *AI Mag. 3*, 2, 23–35.

ORD-HUME, A. W. J. G. 1983. Cogs and crochets: A view of mechanical music. *Early Music 11*, 2, 167–171.

PISZCZALSKI, N., AND GALLER, B. 1979a. Spectral surfaces in performed music. *Comput. Music J. 3*, 1, 18–24.

PISZCZALSKI, N., AND GALLER, B. 1979b. Spectral surfaces in performed music. *Comput. Music J. 3*, 2, 25–27.

PORTNOFF, M. 1976. Implementation of the digital phase vocoder using the fast Fourier transform. *IEEE Trans. Acoust. Speech, Signal Process. ASSP-24*, 3, 243–248.

RADER, G. 1974. A method for composing simple traditional music by computer. *Commun. ACM 17*, 11 (Nov.), 631–638.

RAHN, J. 1980. On some computational models of music theory. *Comput. Music J. 4*, 2, 66–72.

ROADS, C. 1978. Composing grammars. In *Proceedings of the 1977 International Computer Music Conference*, C. Roads, Ed. Computer Music Association, San Francisco.

ROADS, C. 1979. Grammars as representations for music. *Comput. Music J. 3*, 1, 48–55. Revised and updated in *Foundations of Computer Music*, C. Roads and J. Strawn, Eds. MIT Press, Cambridge, Mass., 1985.

ROADS, C. 1980. Interview with Marvin Minsky. *Comput. Music J. 4*, 3, 25–39.

ROADS, C. 1981. An intelligent composer's assistant. M.I.T. Experimental Music Studio, Cambridge, Mass.

ROADS, C. 1983a. Interactive orchestration based on score analysis. In *Proceedings of the 1983 International Computer Music Conference* (Venice, Italy), J. Strawn and T. Blum, Eds. Computer Music Association, San Francisco.

ROADS, C. 1983b. A report on SPIRE: An interactive audio processing environment. *Comput. Music J. 7*, 2, 70–74.

ROADS, C. 1984. An overview of music representations. In *Musical Grammars and Computer Analysis*, M. Baroni and L. Callegari, Eds. Leo S. Olschki Editore, Florence, Italy.

ROADS, C. 1985a. *Composers and the Computer.* William A. Kaufmann, Los Altos, Calif.

ROADS, C. 1985b. Improvisation with George Lewis. In *Composers and the Computer*, C. Roads, Ed. William A. Kaufmann, Los Altos, Calif.

ROADS, C. 1986. The Tsukuba musical robot. *Comput. Music J. 10*, 1.

ROADS, C. N.d. *A History of Computer Music.* William A. Kaufmann, Los Altos, Calif. In press.

ROADS, C., AND STRAWN, J., EDS. 1985. *Foundations of Computer Music.* MIT Press, Cambridge, Mass.

ROADS, C., AND STRAWN, J. N.d. *Computer Music Tutorial.* MIT Press, Cambridge, Mass. In press.

RODET, X., POTARD, Y., AND BARRIÈRE, J. B. 1984. The Chant project: From synthesis of the singing voice to synthesis in general. *Comput. Music J. 8*, 3, 15–31.

ROTHGEB, J. 1968. Harmonizing the unfigured bass: A computational Study. Ph.D. dissertation, Dept. of Music, Yale Univ., New Haven. Conn.

ROTHGEB, J. 1980. Simulating musical skills by digital computer. *Comput. Music J. 4*, 2, 36–40.

ROWE, N. 1975. Machine perception of musical rhythm. B.S. thesis. Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, Mass.

SANDEWALL, E. 1978. Programming in an interactive environment: The "LISP" experience. *ACM Comput. Surv. 10*, 1 (Mar.), 36–71.

SCHANK, R., AND ABLESON, H. 1977. *Scripts, Plans, Goals, and Understanding.* Erlbaum, Hillsdale, N.J.

SCHOTTSTAEDT, B. 1983. Pla: A composer's idea of a language. *Comput. Music J. 7*, 1, 11–20.

SHIPMAN, D. 1983. SpireX: Statistical analysis in the Spire acoustic/phonetic workstation. Paper presented at the IEEE Conference on Acoustics, Speech, and Signal Processing (Boston, April).

SIMON, E. 1960. *Mechanische Musikinstrumente früherer Zeiten und ihre Musik.* Breitkopf und Härtel, Wiesbaden, West Germany.

SIMON, H., AND SUMNER, R. 1968. Pattern in music. In *Formal Representations of Human Judgement*, B. Kleinmutz, Ed. Wiley, New York.

SMITH, L. 1973. Editing and printing music by computer. *J. Music Theor. 17*, 292–309.

SMOLIAR, S. 1971. A parallel processing model of musical structures. AI-TR-91. Artificial Intelligence Lab., M.I.T., Cambridge, Mass.

SMOLIAR, S. 1974. Process structuring and music theory. *J. Music Theor. 18*, 2 (Summer), 308–336.

SMOLIAR, S. 1980. A computer aid for Schenkerian analysis. *Comput. Music J. 4*, 2, 41–59.

SNELL, J. 1979. Design for a formal system for deriving tonal music. M.A. thesis. Dept. of Music, State Univ. of New York at Binghamton, Binghamton, N.Y.

STEELE, G. 1980. The definition and implementation of a programming language based on constraints. AI-TR-595. Artificial Intelligence Lab., M.I.T., Cambridge, Mass.

STEELS, L. 1979. Reasoning modeled as a society of communicating experts. AI-TR-542. Artificial Intelligence Lab., M.I.T., Cambridge, Mass.

STEELS, L. 1982. Constraints as consultants. In *Proceedings of the 1982 European Conference on Artificial Intelligence.* Univ. of Kaiserslautern, Kaiserslautern, West Germany.

STOCKHAUSEN, K. 1978. Elektronische Musik und Automatik. In *Texte zur Musik: 1963–1970.* DuMont Schauberg, Cologne, West Germany.

STRAWN, J. 1980. Approximation and syntactic analysis of amplitude and frequency functions for digital synthesis. *Comput. Music J. 4*, 3, 3–23.

STRAWN, J. ED. 1985a. *Digital Audio Signal Processing: An Anthology.* William Kaufmann, Los Altos, Calif.

STRAWN, J. ED. 1985b. *Digital Audio Engineering: An Anthology.* William Kaufmann, Los Altos, Calif.

STRAWN, J. 1985c. Modelling transitions in musical instruments. Ph.D. dissertation. Center for Com-

puter Research in Music and Acoustics, Stanford Univ., Stanford, Calif.

SUSSMAN, G., AND STEELE, G. 1981. Constraints: A language for expressing almost-hierarchical descriptions. A.I. Memo 502A. Artificial Intelligence Lab., M.I.T., Cambridge, Mass. Reprinted in *Artif. Intell. 14*, 1–39.

ULRICH, W. 1977. The analysis and synthesis of jazz by computer. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence.* Morgan Kaufman, Los Altos, Calif.

VARÈSE, E. 1971. The liberation of sound. In *Perspectives on American Composers*, B. Boretz and E. Cone, Eds. Norton, New York.

VERCOE, B. 1984. The synthetic performer in the context of live performance. In *Proceedings of the International Computer Music Conference, 1984*, W. Buxton, Ed. Computer Music Association, San Francisco.

WINOGRAD, T. 1968. Linguistics and the computer analysis of tonal harmony. *J. Music Theor. 12*, 2–49.

WINOGRAD, T. 1973. A procedural model of language understanding. In *Computer Models of Thought and Language*, R. Schank and K. Colby, Eds. Freeman, San Francisco.

WINOGRAD, T. 1977. Five lectures on artificial intelligence. In *Linguistic Structures Processing*, A. Zampoli, Ed. North-Holland, Amsterdam.

WINSTON, P. 1975. Learning structural descriptions from examples. In *The Psychology of Computer Vision.* McGraw-Hill, New York.

WINSTON, P. 1979. Learning by creating and justifying transfer frames. In *Artificial Intelligence: An MIT Perspective*, vol. 1, P. Winston, Ed. MIT Press, Cambridge, Mass.

WINSTON, P. 1980. Learning and reasoning by analogy. *Commun. ACM 23*, 12 (Dec.), 609–703.

WINSTON, P. 1981. *LISP.* Addison-Wesley, Reading, Mass.

WINSTON, P. 1984. *Artificial Intelligence*, 2nd ed. Addison-Wesley, Reading, Mass.

WOODS, W. 1981a. Procedural semantics as a theory of meaning. Rep. 4627, Bolt, Beranek and Newman, Cambridge, Mass.

WOODS, W. 1981b. Research in knowledge representation for natural language understanding. Rep. 4785, Bolt, Beranek and Newman, Cambridge, Mass.

XENAKIS, I. 1971. *Formalized Music.* Indiana University Press, Bloomington, Ind.

YAVELOW, C. 1985. Music software for the Apple Macintosh. *Comput. Music J. 9*, 3.

ZARIPOV, R. 1969. Cybernetics and music. *Perspectives New Music 7*, 2 (Spring/Summer), 115–154.

Received March 1984; final revision accepted July 1985.